PATENT **MS174297.01/MSFTP245US**

CERTIFICATE OF FACSIMILE TRANSMISSION

I hereby certify that this correspondence (along with any paper referred to as being attached or enclosed) is being faxed to 571-273-8300 on the date shown below to **Mail Stop Appeal Brief – Patents**, Commissioner for Patents, P.O. Box 1450, Alexandria, Virginia 22313-1450.

Date: 10-5-05 Christina M. Padamonsky

RECEIVED
CENTRAL FAX CENTER
OCT 0 5 2005

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re patent application of:			
Applicants(s):	Gopala Krishna R. Kakivaya, *et al.*	Examiner:	Nilesh R. Shah
Serial No:	09/894,700	Art Unit:	2195
Filing Date:	June 28, 2001		

Title: CLASS INITIALIZATION METHOD SEMANTICS

Mail Stop Appeal Brief – Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

APPEAL BRIEF

Dear Sir:

Applicant's representative respectfully submits this brief in connection with an appeal of the above-identified patent application. A credit card payment form is filed concurrently herewith in connection with all fees due regarding this appeal brief. In the event any additional fees may be due and/or are not covered by the credit card, the Commissioner is authorized to charge such fees to Deposit Account No. 50-1063 [MSFTP245US].

10/07/2005 TL0111 00000012 09894700
01 FC:1402 500.00 OP

I. Real Party in Interest (37 C.F.R. §41.37(c)(1)(i))

The real party in interest in the present appeal is Microsoft Corporation, the assignee of the present application.

II. Related Appeals and Interferences (37 C.F.R. §41.37(c)(1)(ii))

Appellant, appellant's legal representative, and/or the assignee of the present application are not aware of any appeals or interferences which may be related to, will directly affect, or be directly affected by or have a bearing on the Board's decision in the pending appeal.

III. Status of Claims (37 C.F.R. §41.37(c)(1)(iii))

Claims 1-34 stand rejected by the Examiner. The rejection of claims 1-34 is being appealed.

IV. Status of Amendments (37 C.F.R. §41.37(c)(1)(iv))

No amendments have been entered after the Final Office Action.

V. Summary of Claimed Subject Matter (37 C.F.R. §41.37(c)(1)(v))

A. Independent Claim 1

Independent claim 1 recites a system for mitigating problems associated with automatic execution of initialization code, the system comprising: an initialization method activator that calls a class initialization method at a pre-determined execution point; and a deadlock analyzer that determines whether running the class initialization method will produce a deadlock.. (*See e.g.*, page 9, lines 6-28)

B. Independent Claim 15

Independent claim 15 recites a computer readable medium containing computer executable components of a system for mitigating problems associated with automatic execution of initialization code, the system comprising: an initialization method activating component that calls the class initialization method at a pre-determined execution point; and a deadlock analyzer

that determines whether running the class initialization method will produce a deadlock.. (*See e.g.*, page 9, lines 6-28)

C. **Independent Claim 16**

Independent claim 16 recites a computer readable medium containing computer executable components of a system for mitigating problems associated with automatic execution of initialization code, the system comprising: a semantic analyzing component that determines a semantic type associated with the initialization method; a domain uniqueness analyzing component that determines a uniqueness type associated with one or more application domains with which the class will interact; a deadlock analyzing component that determines whether calling the initialization method will create a deadlock and resolves the deadlock; and an initialization method activating component that calls the initialization method at a pre-determined execution point, where the pre-determined execution point depends on, at least in part, the semantic type and the domain uniqueness (*See e.g.*, page 9, line 6 – page 10, line 29)

D. **Independent Claim 17**

Independent claim 17 recites a method for mitigating problems associated with automatic execution of class initialization code, the method comprising: determining whether a class has an initializing method; determining when the initializing method should be run; associating initialization check code with one or more components associated with a runtime, the check code determines whether a class is initialized; determining whether calling the initializing method will generate a deadlock and if calling the initializing method will generate a deadlock, resolving the deadlock; and calling the class initializing method. (*See e.g.*, page 11, line 13 – page 14, line 18)

E. **Independent Claim 27**

Independent claim 27 recites a computer readable medium containing computer executable instructions that perform a method for mitigating problems associated with automatic execution of class initialization code, the method comprising: determining whether a class has an initializing method; determining when the initializing method should be run; inserting initialization check code into one or more components associated with a runtime; determining whether calling the initializing method will generate a deadlock and if calling the initializing

method will generate a deadlock, resolving the deadlock; and calling the class initializing method. (*See e.g.*, page 11, line 13 – page 14, line 18)

F. Independent Claim 28

Independent claim 28 recites a method for mitigating problems associated with automatic execution of class initialization code, the method comprising: determining whether a class has an initializing method; determining when the initializing method should be run, where determining when the initializing method should be run comprises: analyzing semantic information associated with the initializing method, where the semantic information comprises an identifier that identifies whether the initializing method desires "exact" or "before field initialization" behavior; and analyzing domain uniqueness information associated with one or more domains with which the initializing method will interact, where the domain uniqueness information comprises an identifier that identifies whether the initializing method is operating in a "normal" or a "domain neutral" environment; associating initialization check code with one or more components associated with a runtime; determining whether calling the initializing method will generate a deadlock; resolving the deadlock; and calling the class initializing method. (See e.g., page 9, line 6 – page 10, line 29, page 11, line 13 – page 14, line 18)

G. **Independent Claim 31**

Independent claim 31 recites a computer readable medium containing computer executable instructions for performing a method for mitigating problems associated with automatic execution of class initialization code, the method comprising: determining whether a class has an initializing method; determining when the initializing method should be run, where determining when the initializing method should be run comprises: analyzing semantic information associated with the initializing method, where the semantic information comprises an identifier that identifies whether the initializing method desires "exact" or "before field initialization" behavior; and analyzing domain uniqueness information associated with one or more environments with which the initializing method will interact, where the domain uniqueness information comprises an identifier that identifies whether the initializing method will interact with a "normal" or a "domain neutral" environment; associating initialization check code with one or more components associated with a runtime; determining whether calling the initializing method will generate a deadlock; resolving the deadlock; and calling the class initializing method. (See e.g., page 9, line 6 – page 10, line 29, page 11, line 13 – page 14, line 18)

H. **Independent Claim 32**

Independent claim 32 recites a system for mitigating problems associated with automatic execution of class initialization code, the method comprising: means for identifying a constructor associated with a class; means for scheduling the running of the constructor; means for adding code into one or more components generated by a runtime, the code identifies whether a class is initialized; means for detecting deadlocks between constructors; means for resolving deadlocks between constructors; and means for invoking a constructor. (*See e.g.*, page 9, lines 6-28, page 22, line 30 – page 23, line 23)

I. **Independent Claim 33**

Independent claim 33 recites a data packet adapted to be transmitted between two or more components, the data packet comprising: information associated with one or more nodes associated with a wait for graph, where the nodes model one or more threads to be analyzed to determine whether class initialization code will generate a deadlock; and information associated with one or more arcs associated with a wait for graph, where the arcs model one or more wait for relationships between one or more of the nodes. (*See e.g.*, page 6, line 1-11, line 23, page 9, lines 6-28)

J. **Independent Claim 34**

Independent claim 34 recites a data packet adapted to be transmitted between two or more components, the data packet comprising: a first field that holds information concerning the identity of a thread that is attempting to initialize a class; a second field that holds information concerning the identity of one or more threads that are waiting for a class to be initialized; and a third field that holds information concerning the initialization status of a class to facilitate deadlock detection and resolution. (*See e.g.*, page 6, line 1-11, line 23, page 9, lines 6-28)

VI. Grounds of Rejection to be Reviewed (37 C.F.R. §41.37(c)(1)(vi))

A. Whether claims 1-34 are unpatentable under 35 U.S.C. §103(a) over over Bak, *et al.* (U.S. 5,999,732) in view of Yantchev, *et al.*, "Adaptive, low latency, deadlock-free packet routing for networks of processors", IEEE Proceedings, 136, 178-186 (May 1989).

VII. Argument (37 C.F.R. §41.37(c)(1)(vii))

A. **Rejection of Claims 1-34 Under 35 U.S.C. §103(a)**

Claims 1-34 stand rejected under 35 U.S.C. §103(a) as being unpatentable over Bak, *et al.* (U.S. 5,999,732) in view of Yantchev, *et al.*, "Adaptive, low latency, deadlock-free packet routing for networks of processors", IEEE Proceedings, 136, 178-186 (May 1989). This rejection should be withdrawn for at least the following reasons. Bak, *et al.* and Yantchev, *et al.*, alone or in combination, do not teach or suggest each and every limitation of applicants' claimed invention.

> To reject claims in an application under §103, an examiner must establish a *prima facie* case of obviousness. A *prima facie* case of obviousness is established by a showing of three basic criteria. First, there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or to combine reference teachings. Second there must be a reasonable expectation of success. Finally, the prior art reference (or references when combined) ***must teach or suggest all the claim limitations.*** *See* MPEP §706.02(j). The ***teaching or suggestion to make the claimed combination*** and the reasonable expectation of success ***must be found in the prior art and not based on the Applicant's disclosure.*** *See In re Vaeck*, 947 F.2d 488, 20 USPQ2d 1438 (Fed. Cir. 1991) (emphasis added).

The subject invention relates to detection, prevention and resolution of deadlocks associated with execution of class initialization code. In particular, independent claims 1 and 15 recite *an initialization method activator that calls a class initialization method at a pre-determined execution point and* ***a deadlock analyzer that determines whether running the class initialization method will produce a deadlock.*** The claimed deadlock analyzer performs an analysis prior to running the class initialization method to determine if running the method will produce a deadlock. As conceded in the Office Action, Bak, *et al.* fails to teach or suggest such a deadlock analyzer. Rather, Bak, *et al.* teaches a method for inserting placeholder data in instructions during pre-compilation of class cope if a variable is unavailable. The cited reference is silent regarding detection of deadlocks prior to running class initialization, and in fact does not use the term deadlock anywhere in the patent. Furthermore, contrary to assertions in the Office Action, Yantchev, *et al.* also does not teach or suggest a deadlock analyzer that determines whether running the class initialization method will produce a deadlock. Rather, Yantchev, *et al.* teaches deadlock prevention associated with packet routing communication networks, and does not pertain to execution of class initialization code in any manner. The Office Action states that "the messages are class messages are blocked thus avoiding deadlock" suggesting that routing messages is equivalent to class initialization code. However, Yantchev, *et al.* does not make any mention of class initialization code and only uses the term class in relation to messages and buffers. Specifically, the cited reference discusses classes in terms of message buffer classes that are assigned to messages in a store and forward packet routing algorithm. Additionally, the term class is used in relation packets assigned to classes representing four quandrants in a 2D plane

associated with a 2D array network. Bak, *et al.* relates to techniques for reducing cost of dynamic class loading and initialization checks in compiled code, and in particular techniques for reducing the cost of dynamic class loading, and Yantchev, *et al.* pertains to the provision of deadlock prevention in a packet routing network. Notwithstanding the cited references failing to result in the claimed invention when combined, they relate to different subject matter and provide no motivation to be combined with one another in the manner suggested absent impermissible use of applicants' claimed invention as a 20/20 hindsight based roadmap. Thus, it is submitted that the combination of Bak, *et al.* and Yantchev, *et al.* fails not only to teach or suggest an initialization method activator that calls a class initialization method at a pre-determined execution point and a deadlock analyzer that determines whether running the class initialization method will produce a deadlock, but also that the combination of Bak, *et al.* and Yantchev, *et al.* is impermissible.

Independent claim 16 recites ***a semantic analyzing component that determines a semantic type associated with the initialization method*** and ***a deadlock analyzing component*** that ***determines whether calling the initialization method will create a deadlock and resolves the deadlock***. Contrary to assertions in the Office Action, Bak, *et al.* fails to provide the semantic analyzing component. Rather, Bak, *et al.* provides a routine that analyzes a class field wherein the routine *resolves the location of a field in memory and the class to which it belongs.* On the other hand, the claimed invention utilizes a semantic analyzer to determine a semantic type associated with the initialization method that in turn ***determines when the initialization method should run***. Furthermore as discussed *supra* with respect to independent claim 1, Yantchev, *et al.* fails to teach or suggest the deadlock analyzing component recited in applicants' claimed invention - Yantchev, *et al.* fails to provide a deadlock analyzing component that can determine not only whether the invocation of an initialization method will create a deadlock, but that can also resolve the deadlock. Moreover, as discussed above, Yantchev, *et al.* makes no mention of class initialization, much less determining and resolving deadlocks thereof. Thus, the combination of Bak, *et al.* and Yantchev, *et al.* fails to teach or suggest a semantic analyzing component that determines a semantic type associated with the initialization method and a deadlock analyzing component that determines whether calling the initialization method will create a deadlock and resolves the deadlock as in the claimed invention.

Independent claims 17 and 27 recite ***determining whether calling the initializing method will generate a deadlock and if calling the initializing method will generate a deadlock, resolving the deadlock***. As conceded in the Office Action, Bak, *et al.* fails to provide the recited limitation. The Office Action relies upon Yantchev, *et al.* to cure the deficiencies of Bak, *et al.* However, as stated *supra*, Yantchev, *et al.* is directed to analysis of deadlocks when routing packets on a computer network rather than determining whether calling the initializing method will generate a deadlock, and if so resolving the determined deadlock. Accordingly, the combination of Bak, *et al.* and Yantchev, *et al.* does not teach or suggest the novel features of applicants' claimed invention as recited in independent claims 17 and 27.

Independent claims 28 and 31 recite ***analyzing semantic information associated with the initializing method, where the semantic information comprises an identifier that identifies whether the initializing method desires "exact" or "before field initialization" behavior; and determining whether calling the initializing method will generate a deadlock; resolving the deadlock***. As discussed above with respect to independent claim 16, Bak, *et al.* fails to teach or suggest analyzing semantic information associated with an initialization method. Moreover, Bak, *et al.* also does not teach that the semantic information specifies "exact" or "before field initialization" behavior for initializing the method. Furthermore as noted *supra* with respect to independent claim 1, Yantchev, *et al.* fails to teach determining whether calling the initializing method will generate a deadlock, as well as resolving the deadlock. Therefore, the combination of Bak, *et al.* and Yantchev, *et al.* fails to teach or suggest analyzing semantic information associated with the initializing method, where the semantic information comprises an identifier that identifies whether the initializing method desires "exact" or "before field initialization" behavior; and determining whether calling the initializing method will generate a deadlock; resolving the deadlock.

Independent claim 32 recites ***means for identifying a constructor associated with a class*** and ... ***means for detecting deadlocks between constructors***. As the Office Action acknowledges, Bak, *et al.* does not teach a means for detecting deadlocks between constructors, and thus the Examiner offers Yantchev, *et al.* to rectify this deficiency. However, as noted above, Yantchev, *et al.* fails to disclose identification of deadlocks between constructors associated with a class, but rather, teaches deadlock prevention associated with packet routing

communication networks. Therefore, the combination of Bak, *et al.* and Yantchev, *et al.* does not teach or suggest the entirety of independent claim 32.

Independent claim 33 recites ***a data packet adapted to be transmitted between two or more components, the data packet comprising information associated with one or more nodes associated with a wait for graph, where the nodes model one or more threads to be analyzed to determine whether class initialization code will generate a deadlock.*** Although both Bak, *et al* and Yantchev, *et al* teach data packets, neither discloses a data packet containing information associated with a wait for graph. Bak, *et al.* merely teaches a data signal may be the computer readable storage medium. Further, Yantchev, *et al.* discloses a data packet in terms of network routing and separately provides a wait for graph to identify packet communication deadlocks. Yantchev, *et al.* however fails to disclose that the data packets containing information are associated with the wait for graph. Thus, it is submitted the combination of Bak, *et al.* and Yantchev, *et al.* fails to teach or suggest the limitations recited in independent claim 33.

Independent claim 34 recites ***a data packet adapted to be transmitted between two or more components, the data packet comprising: a first field that holds information concerning the identity of a thread that is attempting to initialize a class; a second field that holds information concerning the identity of one or more threads that are waiting for a class to be initialized; and a third field that holds information concerning the initialization status of a class to facilitate deadlock detection and resolution.*** Bak *et al.* and Yantchev *et al.* fail to describe the contents of any fields of a data packet. More specifically, neither Bak, *et al.* nor Yantchev, *et al.* disclose the contents of the three fields as recited in the subject claim. Therefore, the combination of Bak, *et al.* and Yantchev, *et al.* fails to teach or suggest a data packet adapted to be transmitted between two or more components, the data packet comprising: a first field that holds information concerning the identity of a thread that is attempting to initialize a class; a second field that holds information concerning the identity of one or more threads that are waiting for a class to be initialized; and a third field that holds information concerning the initialization status of a class to facilitate deadlock detection and resolution.

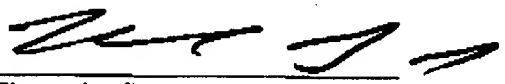
In view of at least the above, it is respectfully submitted that Bak, *et al.* and Yantchev, *et al.*, alone or in combination, do not make obvious the subject invention as recited in independent claims 1, 15, 16, 17, 27, 28, 31, 32, 33, and 34 (and claims 2-14, 18-26, 29, and 30 which respectively depend there from). Accordingly, withdrawal of this rejection is respectfully requested.

C. **Conclusion**

For at least the above reasons, the claims currently under consideration are believed to be patentable over the cited references. Accordingly, it is respectfully requested that the rejections of claims 1-34 be reversed.

If any additional fees are due in connection with this document, the Commissioner is authorized to charge those fees to Deposit Account No. 50-1063 [MSFTP245US].

Respectfully submitted,
AMIN & TUROCY, LLP



Himanshu S. Amin
Reg. No. 40,894

AMIN & TUROCY, LLP
24th Floor, National City Center
1900 East 9th Street
Telephone: (216) 696-8730
Facsimile: (216) 696-8731

VIII. Claims Appendix (37 C.F.R. §41.37(c)(1)(viii))

1. A system for mitigating problems associated with automatic execution of initialization code, the system comprising:

an initialization method activator that calls a class initialization method at a pre-determined execution point; and

a deadlock analyzer that determines whether running the class initialization method will produce a deadlock.

2. The system of claim 1, where the initialization method activator checks whether a class is initialized and, if the class is not initialized, calls the class initialization method.

3. The system of claim 2, where the deadlock analyzer determines whether calling the class initialization method will generate a deadlock.

4. The system of claim 3, where the pre-determined execution point is at least one of one of a caller's just in time compilation time, a callee's just in time compilation time, an initial field access time, an initial method access time, an initial static field access time and a first access of pre-compiled code where no just in time compilation occurs.

5. The system of claim 1, where the initialization method detector associates initialization check code with one or more components associated with a runtime environment, where the initialization check code determines whether a class has been initialized.

6. The system of claim 5, where the initialization check code determines whether calling the class initialization method will generate a deadlock, and if a deadlock will be generated, to resolve the deadlock.

7. The system of claim 6, where the initialization check code is run at one of a caller's just in time compilation time, a callee's just in time compilation time, a first field access time, a first

method access time, a first static field access time and a first access of pre-compiled code where no just in time compilation occurs.

8. The system of claim 1, where the deadlock analyzer analyzes a wait for graph.

9. The system of claim 8 where the deadlock analyzer resolves a deadlock associated with running the class initialization method.

10. The system of claim 9, where the deadlock analyzer adds and/or removes one or more nodes and/or arcs from the wait for graph.

11. The system of claim 8, further comprising a semantic analyzer that analyzes a semantic type associated with the class initialization method, where the semantic analyzer provides information concerning a desired initialization check time to the initialization method activator.

12. The system of claim 11, where the semantic type is one of "exact" and "before field initialization".

13. The system of claim 11, further comprising a domain uniqueness analyzer that analyzes the uniqueness of one or more domains with which the class initialization method and/or the class will interact, where the uniqueness analyzer provides information concerning a desired initialization check time to the initialization method activator.

14. The system of claim 13, where the domain uniqueness is one of "normal" and "domain neutral".

15. A computer readable medium containing computer executable components of a system for mitigating problems associated with automatic execution of initialization code, the system comprising:

an initialization method activating component that calls the class initialization method at a pre-determined execution point; and

a deadlock analyzer that determines whether running the class initialization method will produce a deadlock.

16. A computer readable medium containing computer executable components of a system for mitigating problems associated with automatic execution of initialization code, the system comprising:

a semantic analyzing component that determines a semantic type associated with the initialization method;

a domain uniqueness analyzing component that determines a uniqueness type associated with one or more application domains with which the class will interact;

a deadlock analyzing component that determines whether calling the initialization method will create a deadlock and resolves the deadlock; and

an initialization method activating component that calls the initialization method at a pre-determined execution point, where the pre-determined execution point depends on, at least in part, the semantic type and the domain uniqueness.

17. A method for mitigating problems associated with automatic execution of class initialization code, the method comprising:

determining whether a class has an initializing method;

determining when the initializing method should be run;

associating initialization check code with one or more components associated with a runtime, the check code determines whether a class is initialized;

determining whether calling the initializing method will generate a deadlock and if calling the initializing method will generate a deadlock, resolving the deadlock; and

calling the class initializing method.

18. The method of claim 17, where determining when the initializing method should be run comprises:

analyzing semantic information associated with the initializing method.

19. The method of claim 18, where the semantic information comprises an identifier that identifies whether the initializing method desires "exact" or "before field initialization" behavior.

20. The method of claim 19, where determining when the initializing method should be run further comprises analyzing domain uniqueness information associated with one or more domains with which the class initialization code will interact.

21. The method of claim 20, where the domain uniqueness information comprises an identifier that identifies whether the initializing method is associated with a "normal" or a "domain neutral" environment.

22. The method of claim 17, where determining whether calling the initializing method will generate a deadlock comprises:

attempting to acquire an initialization lock associated with the class to be initialized, and if the initialization lock cannot be acquired, identifying a holding thread that is holding the initialization lock;

locating a node associated with the holding thread, where the node is located in a wait for graph; and

analyzing the wait for graph to determine whether a deadlock exists.

23. The method of claim 22, where analyzing the wait for graph to determine whether a deadlock exists comprises traversing the wait for graph starting at the node associated with the holding thread and determining whether a cycle is detected in the wait for graph.

24. The method of claim 23, where resolving the deadlock comprises:

acquiring a lock associated with the wait for graph;

if a detecting thread that identifies the deadlock previously added one or more arcs and/or nodes to the wait for graph, removing the one or more arcs and/or nodes from the wait for graph;

releasing the lock associated with the wait for graph; and

the detecting thread interacting with the class as though the class was initialized.

25. The method of claim 17, where determining whether calling the initializing method will generate a deadlock comprises:

an acquiring thread attempting to acquire an initialization lock associated with the class to be initialized, and, if the lock can not be acquired, determining whether there is a thread waiting for the acquiring thread to complete its processing and release the initialization lock.

26. The method of claim 25, where resolving the deadlock comprises:

if it was determined that there was a thread waiting for the acquiring thread to complete its processing and release the initialization lock, then returning and interacting with the partially initialized state of the class as though the class were initialized, otherwise, blocking until the class becomes initialized.

27. A computer readable medium containing computer executable instructions that perform a method for mitigating problems associated with automatic execution of class initialization code, the method comprising:

determining whether a class has an initializing method;

determining when the initializing method should be run;

inserting initialization check code into one or more components associated with a runtime;

determining whether calling the initializing method will generate a deadlock and if calling the initializing method will generate a deadlock, resolving the deadlock; and

calling the class initializing method.

28. A method for mitigating problems associated with automatic execution of class initialization code, the method comprising:

determining whether a class has an initializing method;

determining when the initializing method should be run, where determining when the initializing method should be run comprises:

analyzing semantic information associated with the initializing method, where the semantic information comprises an identifier that identifies whether the initializing method desires "exact" or "before field initialization" behavior; and

analyzing domain uniqueness information associated with one or more domains with which the initializing method will interact, where the domain uniqueness information comprises an identifier that identifies whether the initializing method is operating in a "normal" or a "domain neutral" environment;

associating initialization check code with one or more components associated with a runtime;

determining whether calling the initializing method will generate a deadlock;

resolving the deadlock; and

calling the class initializing method.

29. The method of claim 28, where determining whether calling the initializing method will generate a deadlock comprises:

attempting to acquire an initialization lock associated with the class to be initialized, and if the initialization lock cannot be acquired, identifying a holding thread that is holding the initialization lock;

locating a node associated with the holding thread, where the node is located in a wait for graph; and

analyzing the wait for graph to determine whether a deadlock exists, where analyzing the wait for graph to determine whether a deadlock exists comprises:

traversing the wait for graph starting at the node associated with the holding thread and determining whether a cycle is detected in the wait for graph.

30. The method of claim 29, where resolving the deadlock comprises:

acquiring a lock associated with the wait for graph;

if the thread that identifies the deadlock previously added one or more arcs and/or nodes to the wait for graph, removing the one or more arcs and/or nodes from the wait for graph;

releasing the lock associated with the wait for graph; and

in the thread that detected that it could not initialize the class because a deadlock existed with another thread that was initializing the class, interacting with the class as though it was initialized.

31. A computer readable medium containing computer executable instructions for performing a method for mitigating problems associated with automatic execution of class initialization code, the method comprising:

determining whether a class has an initializing method;

determining when the initializing method should be run, where determining when the initializing method should be run comprises:

analyzing semantic information associated with the initializing method, where the semantic information comprises an identifier that identifies whether the initializing method desires "exact" or "before field initialization" behavior; and

analyzing domain uniqueness information associated with one or more environments with which the initializing method will interact, where the domain uniqueness information comprises an identifier that identifies whether the initializing method will interact with a "normal" or a "domain neutral" environment;

associating initialization check code with one or more components associated with a runtime;

determining whether calling the initializing method will generate a deadlock;

resolving the deadlock; and

calling the class initializing method.

32. A system for mitigating problems associated with automatic execution of class initialization code, the method comprising:

means for identifying a constructor associated with a class;

means for scheduling the running of the constructor;

means for adding code into one or more components generated by a runtime, the code identifies whether a class is initialized;

means for detecting deadlocks between constructors;

means for resolving deadlocks between constructors; and

means for invoking a constructor.

33. A data packet adapted to be transmitted between two or more components, the data packet comprising:

information associated with one or more nodes associated with a wait for graph, where the nodes model one or more threads to be analyzed to determine whether class initialization code will generate a deadlock; and

information associated with one or more arcs associated with a wait for graph, where the arcs model one or more wait for relationships between one or more of the nodes.

34. A data packet adapted to be transmitted between two or more components, the data packet comprising:

a first field that holds information concerning the identity of a thread that is attempting to initialize a class;

a second field that holds information concerning the identity of one or more threads that are waiting for a class to be initialized; and

a third field that holds information concerning the initialization status of a class to facilitate deadlock detection and resolution.

IX. Evidence Appendix (37 C.F.R. §41.37(c)(1)(ix))

None.

X. Related Proceedings Appendix (37 C.F.R. §41.37(c)(1)(x))

None.